HTTP协议

2022年4月26日 15:41

HTTP1.0

Key Features

a. 引入了请求头和响应头
b. 引入了版本信息
c. 它允许对每个 TCP 连接进行单个请求/响应
d. 状态码用于指示成功的请求和指示传输错误。
e. Content-Type 标头使发送非纯 HTML 文件成为可能，包括脚本和媒体。

HTTP/1.1

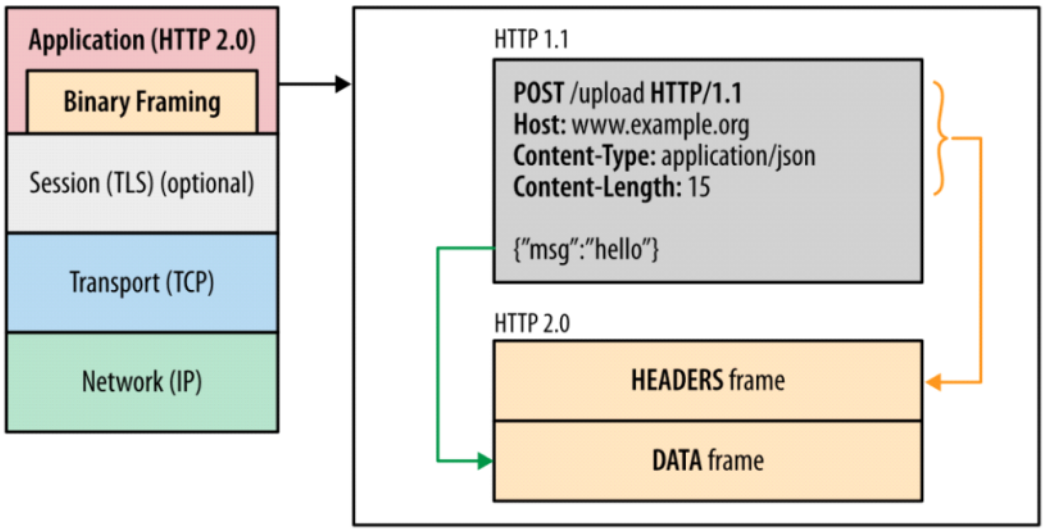
Key Features

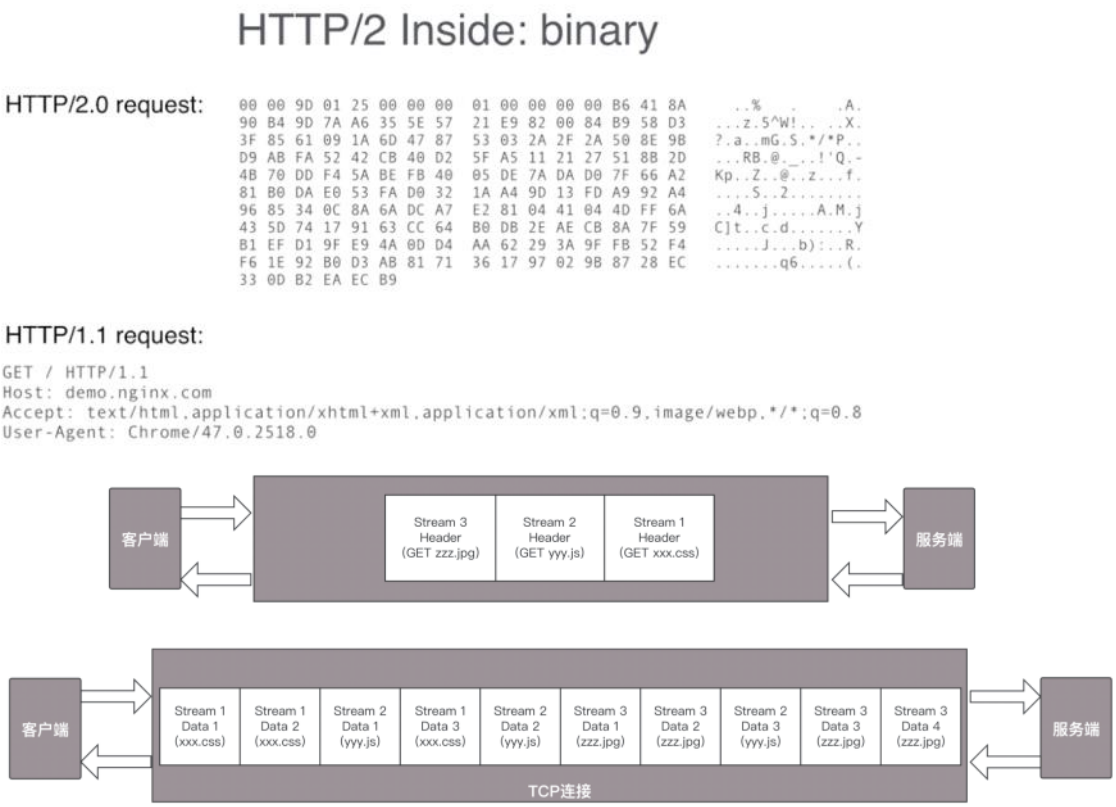
a. 它允许每个 TCP 连接有多个请求/响应。 keep-alive头可以建立持久连接
b. Upgrade 标头用于指示客户端的偏好，如果服务器认为合适，则可以切换到更偏好的协议。
c. HTTP/1.1提供了对块传输的支持，这些块传输允许作为块动态流动内容流，并在消息主体之后发送其他标头。
    i. 优点：对于动态生成的内容来说，在内容创建完之前是不可知的。分块传输能很好的解决这种情形的传输编问题，它允许服务器在最后发送消息头字段。这样对于传输数据时，不需要等到所有的内容都计算好后在进行。
d. 流水线：第二个请求在对第一个请求的响应得到充分服务之前发送
e. 内容协商：客户端和服务器之间的交换以确定媒体类型
f. 缓存控制：用于在请求和响应中指定缓存策略

HTTP/2

Key Features

a. 引入了服务端推送的概念，预判资源进行推送
b. 引入了多路复用的概念，该概念在没有线头阻塞的情况下交错请求和响应，并通过单个 TCP 连接执行此操作。
c. 它是一个二进制协议，即仅以0s的形式和1s的二进制命令在电线上传输。二进制框架层将消息划分为基于其类型 - 数据或标头隔离的帧。此功能大大提高了安全性，压缩和多路复用方面的效率。





d. HTTP/2 使用 HPACK 标头压缩算法，该算法对 CRIME 等攻击具有弹性，并利用静态 Huffman 编码。

e. HTTP 2.0 成功解决了 HTTP 1.1 的队首阻塞问题，同时，也不需要通过 HTTP 1.x 的 pipeline 机制用多条 TCP 连接来实现并行请求与响应；

   i. 应用层的队首阻塞(HTTP/1.1-based head of line blocking): HTTP/1.1可以使用多个TCP连接，但对连接数依然有限制，一次请求要等到连接中其他请求完成后才能开始(Pipeline机制也没能解决好这个问题)，所以没有空闲连接的时候请求被阻塞，这是应用层的阻塞。 HTTP/2底层使用了一个TCP连接，上层虚拟了stream，HTTP请求跟stream打交道，无需等待前面的请求完成，这确实解决了应用层的队首阻塞问题。

   ii. 传输层的队首阻塞(TCP-based head of line blocking): 正如文中所述，TCP的应答是严格有序的，如果前面的包没到，即使后面的到了也不能应答，这样可能会导致后面的包被重传，窗口被“阻塞”在队首，这就是传输层的队首阻塞。 不管是HTTP/1.1还是HTTP/2，在传输层都是基于TCP，那么TCP层的队首阻塞问题都是存在的(只能由HTTP/3(based on QUIC)来解决了)，另外，HTTP/2用了单个TCP连接，那么在丢包率严重的场景，表现可能比HTTP/1.1更差。

f. 减少了 TCP 连接数对服务器性能的影响，同时将页面的多个数据 css、js、 jpg 等通过一个数据链接进行传输，能够加快页面组件的传输速度。

QUIC 协议的“城会玩”

机制一：自定义连接机制

a. TCP连接在移动互联网中的问题

   在移动互联情况下，当手机信号不稳定或者在 WIFI 和 移动网络切换时，都会导致重连，从而进行再次的三次握手，导致一定的时延。

b. QUIC

   这在 TCP 是没有办法的，但是基于 UDP，就可以在 QUIC 自己的逻辑里面维护连接的机制，不再以

四元组标识，而是以一个 64 位的随机数作为 ID 来标识，而且 UDP 是无连接的，所以当 IP 或者端口变化的时候，只要 ID 不变，就不需要重新建立连接。

机制二：自定义重传机制

a. TCP RTT采样不精确问题

发送一个包，序号为 100，发现没有返回，于是再发送一个 100，过一阵返回一个 ACK101。这个时候客户端知道这个包肯定收到了，但是往返时间是多少呢？是 ACK 到达的时间减去后一个 100 发送的时间，还是减去前一个 100 发送的时间呢？事实是，第一种算法把时间算短了，第二种算法把时间算长了。

b. QUIC解决方案

任何一个序列号的包只发送一次，下次就要加一了。例如，发送一个包，序号是 100，发现没有返回；再次发送的时候，序号就是 101 了；如果返回的 ACK 100，就是对第一个包的响应。如果返回 ACK 101 就是对第二个包的响应，RTT 计算相对准确。

c. 怎么知道包 100 和包 101 发送的是同样的内容呢?

Offset

QUIC 既然是面向连接的，也就像 TCP 一样，是一个数据流，发送的数据在这个数据流里面有个偏移量 offset，可以通过 offset 查看数据发送到了哪里，这样只要这个 offset 的包没有来，就要重发；如果来了，按照 offset 拼接，还是能够拼成一个流。

机制三：无阻塞的多路复用

a. HTTP 2.0 的问题

因为 HTTP 2.0 也是基于 TCP 协议的，TCP 协议在处理包时是有严格顺序的。

当其中一个数据包遇到问题，TCP 连接需要等待这个包完成重传之后才能继续进行。虽然 HTTP 2.0 通过多个 stream，使得逻辑上一个 TCP 连接上的并行内容，进行多路数据的传输，然而这中间并没有关联的数据。一前一后，前面 stream 2 的帧没有收到，后面 stream 1 的帧也会因此阻塞。

机制四：自定义流量控制

a. 滑动窗口

QUIC 的窗口是适应自己的多路复用机制的，不但在一个连接上控制窗口，还在一个连接中的每个 stream 控制窗口。